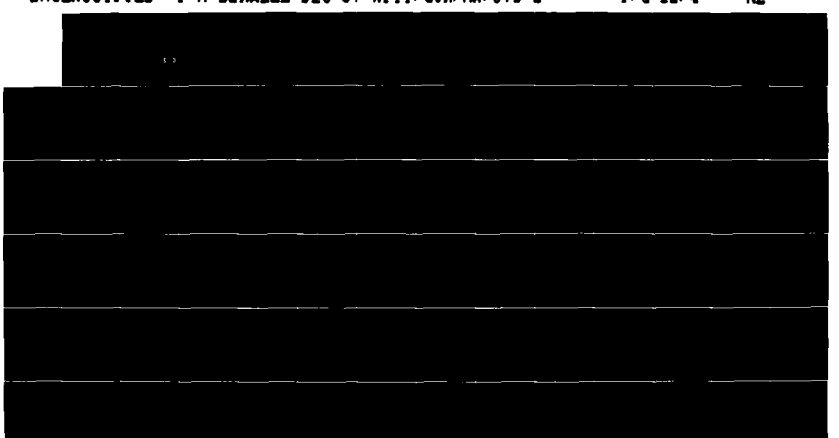
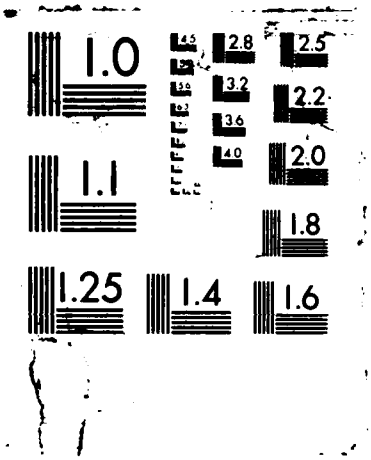
AD-A189 598 PERFORMA: A PERSONAL INFLUENCE DIAGRAM SYSTEM FOR DECISION ANALYSIS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/1

UNCLASSIFIED T M BURWELL DEC 87 AFIT/GOR/MA/87D-2 F/G 12/4 NL



END
DATE FILMED
8-88


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

AFIT/GOR/MA/87D-2

PERFORMA
A PERSONAL INFLUENCE DIAGRAM SYSTEM FOR DECISION ANALYSIS

THESIS

Thomas M. Burwell
Captain, USAF

AFIT/GOR/MA/87D-2

Approved for public release; distribution unlimited

AFIT/GOR/MA/87D-2

PERFORMA

A PERSONAL INFLUENCE DIAGRAM SYSTEM FOR DECISION ANALYSIS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Thomas M. Burwell, B.S.

Captain, USAF

December 1987

Preface

The primary objective was to produce a software system for analyzing influence diagrams for the decision analysis community at the Air Force Institute of Technology. There was an immediate demand for the system once it was running. The need for my work greatly motivated me to produce the software and documentation that resulted in the system called PerForma.

Contributions to PerForma came from several sources. I am deeply indebted to my faculty adviser Captain Joseph Tatman for providing the software MacLab..id that forms the basis of PerForma's mathematical functions and for giving me encouragement and assistance in times of need. I thank Lieutenant Colonel Gregory Parnell, my faculty reader, for contributions to the system design and content of the user's manual and tutorial.

In developing and writing the thesis I owe a great deal of gratitude to my family. I wish to thank my wife Gerry Anne for your love and devotion, my little girl Christina for your innocent understanding through this turmoil, and my son Joseph, only three months old as I put these words to paper, for reminding me that life is a gift even under the most difficult circumstances.

Thomas M. Burwell

Table of Contents

List of Figures

Abstract

The major goal of this research is to develop a widely accessible software system for analyzing and solving influence diagrams for the decision analysis community at the U.S. Air Force Institute of Technology.

A software system for influence diagram analysis, PerForma, was developed. The system makes the influence diagram, a powerful modeling framework for both formulation and analysis, easily accessible. XLISP Version 1.7 was used to write the system. PerForma runs on a broad spectrum of personal computers. The Amiga, Atari, Macintosh, IBM PC and PC compatible computers have a fully operational system. The system is cost free and has no distribution restrictions except for commercial use. The user's manual was written with a documented session of system use. A tutorial was written explaining influence diagram modeling and solution procedure. PerForma was validated through its use in decision analysis application theses.

PerForma is a professional product that will continue to make a significant contribution in the research and analysis of the strategic decision.

PERFORMA
A PERSONAL INFLUENCE DIAGRAM SYSTEM FOR DECISION ANALYSIS

I. Introduction

General Background

A recently developed modeling language that is effective for both formulation and analysis is the influence diagram. An influence diagram is a graphical structure for modeling problems containing random variables, decision variables and deterministic functions explicitly revealing probabilistic dependence and the flow of information (10:871). The influence diagram can serve as both a formal problem description that can be analyzed with a computer and an informal representation that can be easily understood by those that are not acquainted with mathematical formulation. This type of formulation has many areas of application including operations research, stochastic control, classical statistics, and artificial intelligence.

The original application of influence diagrams was in operations research, specifically decision analysis. Decision analysis has been developed over the last 25 years as a methodology to help decision makers make strategic decisions. The influence diagram is an aid in formulating the structure of the decision problem under uncertainty and

a precise description of that structure which can be stored and manipulated by a computer (10:871). There are distinct advantages in using an influence diagram. Some advantages are the following: 1) easy identification of the problem variables; 2) clear representation of the interaction between these variables; and 3) concise statement of the problem. One of the most important advantages is the ability to communicate with computers about the decision problem structure.

Computer software currently exists to analyze and solve influence diagrams. One such software system is MacLab..id (13). This software performs the mathematical operations, but the program is nearly impossible to use because it requires an analyst to have a knowledge of how the program works and an ability to program in the LISP programming language.

Specific Problem

The specific problem is that this potentially useful software system is inaccessible because it requires familiarity with LISP and knowledge of the logic details of the program. Decision analysis students at the U.S. Air Force Institute of Technology do not have a software system as an influence diagram solver that has the following characteristics: 1) runs on a broad spectrum of personal and mainframe computers; 2) is inexpensive; and 3) is adaptable for unusual problems and decision analysis research.

Research Objective

The primary objective of the research is to adapt MacLab..id so that a software system for analyzing and solving influence diagrams is widely and easily accessible to the decision analysis community at the U.S. Air Force Institute of Technology.

Subsidiary Objectives

To achieve the above research objective, the following subsidiary objectives were pursued:

1. Determine the system requirements based on interaction with the decision analysis community at the U.S. Air Force Institute of Technology.
2. Understand the basic logic and architecture of MacLab..id.
3. Determine the equipment needed to use the system, such as memory capacity, and processors.
4. Find the appropriate programming language to write the adaptation, and determine the other software requirements that are necessary to use the system. An approach to the problem could be based on XLISP, a conservative dialect of LISP.
5. Design and build adaptation to integrate the existing software system MacLab..id.
6. Validate the system.
7. Develop a user's manual.
8. Write a system tutorial.

Analysis of Literature

Influence Diagram Formulation in Decision Analysis. One form of problem description is the influence diagram. The influence diagram can serve as both a formal problem description that can be solved by a computer and an informal representation that can be easily understood by those that are not acquainted with mathematical formulation (5:721). This type of formulation is used in decision analysis.

Tatman states that the influence diagram formulation is mathematically precise, and because the formulation is mathematically precise, the influence diagram can be used to state and analyze a decision problem (13:2). Influence diagrams display the structure and qualities of the decision problem, and they also serve as the framework for quantitative analysis (13:8). An advantage to using the influence diagram for formulation is that it discourages errors because it makes explicit the essential elements of the model in a graphical framework that is easily understandable.

The influence diagram has evolved within decision analysis, and the development of computer software has made it feasible to formulate and to analyze the decision analysis problem with an influence diagram (13:2). The influence diagram is hierarchical, containing a top level graph with data as the second level. The reason that the computer can be used for analysis is that the computer is

easily programmed to manipulate this graphical structure of data (13:2).

Tatman also makes a comparison between decision trees and influence diagrams for modeling decision problems (13:10). Trees and influence diagrams are different modeling languages, and they focus on different aspects of the problem. Trees focus on the timing of the events in the model. The influence diagrams focus on the relationships between the variables. The most critical aspects in analyzing a problem is the relationship of dependency and information flow which the influence diagram captures (13:10).

Howard and Matheson give further comparison between influence diagrams and decision trees (5:740). They say that some influence diagrams do not have corresponding trees. If an influence diagram having a complex relationship between its variables is translated into a decision tree, then the resulting tree could have some errors in the statement of influences. Therefore trees are limited in their ability to model accurately those critical aspects noted above (5:740).

A new application for the influence diagram has been developed by Kenley (6:1). The influence diagram is a valid representation of both continuous and discrete variables. However, until now, analysis of the discrete variable influence diagram was only possible. He has developed and

implemented influence diagram theory that allows the analysis of decision problems with continuous variables. These developments will widely expand the area of influence diagram application.

Owen gives justification for the use of influence diagrams for the modeling framework of decision problems (9:766). The decision analyst is working for a decision maker by guiding the decision process, and one of the biggest obstacles for both the analyst and the decision maker is the communication process. Owen states that the influence diagram is an effective communication tool because it coincides with the decision maker's intuitive understanding of the word influence (9:766). The decision maker identifies with the concept of an influence when dealing with the variables in the graphical representation of his problem (9:766-767).

An insight into the importance of the influence diagram to the analyst is given by Howard (4:14). The decision analyst has confronted the problem of reducing all the information the decision maker has 'to a form that could meet the rigid tests of explicitness and consistency required by a computer. The influence diagram is a major aid in this transformation . . .' (4:14). There is a danger of misuse by those that do not fully understand the influence diagram, but the influence diagram holds great

promise as a bridge between the analyst and the decision maker (4:14).

Review of Software Systems for Decision Analysis. In determining the best design for the adaptation of the existing software system, other software systems were evaluated. Three commercial software systems are considered in this literature analysis. These software systems are decision analysis aids and perform a variety of functions. The systems are DAVID, Influence Diagram Processing System for the Macintosh, by Ross D. Shachter and the Center for Health Policy Research and Education, Duke University (2), ARBORIST Decision Tree Software by Texas Instruments (1), SMLTREE, The All Purpose Decision Tree Builder, by Jim Hollenberg and the Pratt Medical Group (12).

DAVID, Influence Diagram Processing System for the Macintosh, is a decision analysis system using the language of influence diagrams (2). The analysis capabilities of DAVID seem to be complete for most influence diagram problems. These features include the following: 1) graphical display for constructing and revising the influence diagram graph structure; 2) sensitivity analysis for probabilities, deterministic parameters, risk tolerance, and the value of information; 3) analysis of the value of information; 4) inspecting and revising variable properties and data; 5) automatic dynamic programming; 6) value lottery analysis to compare policies; 7) an option for a functional

specification for distributions; and 8) graphical display of distributions. The system is considered to have a very user-friendly interface which includes an online help facility and interactive error handling. The user's manual also includes a tutorial. However, there are some distinct disadvantages. DAVID has to run on a Macintosh Plus, SE, or Macintosh II computer and requires at least one megabyte of memory and disk space of at least 800K. Thus, even the average Macintosh owner would be excluded from its use, let alone owners of the Amiga, IBM PC and PC compatible computers (2).

Texas Instruments' ARBORIST is also a decision analysis tool, but this tool uses decision trees as its problem formulation and analysis (1). The software system has graphical display of the decision tree, the probability distribution, and the sensitivity analysis. Another feature for information display is the use of multiple windows. The system possesses a very developed user-friendly interface. The documentation for the system is quite extensive. The system is written in a LISP dialect and runs on IBM PC and PC compatible computer systems (1).

SMLTREE is a decision analysis software system which uses the decision tree for formulation and evaluation of problems (13). The system will allow arbitrarily large decision trees with graphics display of the changes made in the process of the analysis. The system has six major

display features: 1) a node menu of the current nodes; 2) a tree display; 3) a variable menu of the defined variables; 4) a graph display for sensitivity analysis; 5) a table menu containing the current table names; and 6) an options menu for controlling the tree creation process. This system has a user-friendly interface, and it runs on IBM PC and PC compatible computer systems (13).

Conclusion of Literature Analysis. Influence diagrams have become an effective means for not only communicating the decision model between the decision analyst and the decision maker but also between the analyst and the computer. This fact is a result of several factors:

1. The influence diagram is hierarchical, containing a top level graph with data as the second level, and the computer readily understands this graphical structure.
2. The structure of the problem expressed in an influence diagram is an intuitive representation to the decision maker.
3. The influence diagram is more precise than the decision tree since errors in the statement of influences can result in a decision tree formulation.

The analysis of the software systems clearly indicates that there is limited capability to analyze an influence diagram formulation with a computer. There is no system

that is available on a broad spectrum of personal and mainframe computers and is adaptable for unusual problems. The analysis of the systems was very valuable in determining the user-friendly features for the system to be developed. Such features will include the following: a software interface, adequate documentation with user's manual, adaptability to special problems, constructing and revising the influence diagram graph structure, and inspecting and revising variable properties and data.

II. Definition of System Requirements

High Level Issues

The higher level issues of system requirements are derived from the objective to have a software system for analyzing influence diagrams that is widely and easily accessible to the decision analysis community at the U.S. Air Force Institute of Technology (AFIT). This intent drives the requirement for portability among computer systems, usability without being a software programmer, and adaptability for special problems.

Since the students have a wide variety of computer resources, the system should be portable between a diverse set of computer systems without distribution or copy restrictions. The software system should be usable on common computer systems such as an Amiga, Atari, Macintosh, IBM PC and PC compatible. Since the common computer in U.S. Government offices is the IBM PC compatible Zenith Z-248, the student will have the opportunity to learn the software and take it to other work environments after graduating. Therefore, the language for the implementation should be a standard language that is available on all these types of computer systems.

The requirement to know software programming should be kept to a minimum, and the basic features of the software system should be usable by a computer novice. Given the

user's manual, the software system capabilities should be available for use, and no deeper understanding of programming languages should be necessary.

The requirement for adaptability is driven by the need to handle special formulation problems. Each problem has its particular characteristics, and the software should be flexible to perform an unusual function. AFIT is a research organization, and for the unique problem the system needs to be adaptable in providing the analysis required by the AFIT user. Therefore, the system must be free of restrictions that would prevent it from being altered.

Low Level Issues

The lower level issues address the specific requirements of the system. The most obvious source of these requirements is the consideration of what is necessary to model an influence diagram in a software system while analyzing it in the decision analysis cycle.

Commands to model the influence diagram in a computer representation will be the basic building commands, the display commands to show information, and the editing commands for node data. The commands to build an influence diagram must deal with the input of the information. The influence diagram creation, node addition, node development, and arc designation are all necessary for the modeling task. The information needs to be displayed to the user to show the progress of the diagram's development or the results of

the diagram's analysis. The display commands should accomplish the following: show the whole diagram (graph structure and data), show the graph structure, and show the information for an individual node. When displaying the individual node data, the probabilities or values should have their own display commands since there is clearly the possibility of having too much information given at any one level (diagram, graph structure, node, or value level). This convention makes the information display more manageable. The editing commands are for the more time consuming tasks such as the entry of probabilities or values. In dealing with larger models, the edit feature will help the user save time in correcting mistakes and altering the data for sensitivity analysis.

Analysis of the influence diagram through the decision analysis cycle will require the fundamental influence diagram transformations in software system command form. The fundamental influence diagram transformations are the following: arc reversal corresponding to Bayes' rule, decision node removal by maximization, and chance node removal by expectation. These commands are the basic ones necessary for influence diagram analysis.

A user interface must be developed to control system use. The interface controls the input of commands and the sequence of operations. Other basic functions of the interface are data entry, data display, and error handling.

III. Methodology

This chapter explains the methodology used to accomplish the objectives. Several areas had to be addressed to meet the objectives, and these areas are explained individually. The methodology in design of the user interface also includes a review of literature.

Software and Hardware Requirements

Software and hardware requirements were determined by analysis of the existing software system MacLab..id. The hardware requirements were also determined by the desired level of system capability.

Since MacLab..id was written in the XLISP dialect, the only software requirement is the XLISP Version 1.7 interpreter. The XLISP language meets the system requirements defined in the previous chapter. No other software requirements exist.

Computer hardware requirements were made considering the existing system and the level of operating capability for the developed system. The existing system required no special hardware to operate. The capability for the new system requires no more hardware than the average computer so that the portability requirement is met. The average computer with a memory capacity of 512K should be able to operate the system. No special processor, such as a Math-Coprocessor, is necessary to use the system.

Designing the User Interface

The design of an interactive system goes beyond intuitive judgments. The design process must be organized along the theory of interactive computer development. This section reviews the literature of the fundamental theory in user interface design. The developer's intentions for the interface design are also explained.

Literature of Friendly Software Design. There are four stages that a user goes through in using a computer. These stages are presented by Norman (8:365-375), and they correspond to a separation of design concerns. The four stages of an interaction are the following:

1. Forming an intention. The user forms a mental goal.
2. Selecting an action. The user reviews possible actions and selects the most appropriate.
3. Executing the action. The user carries out the action using the computer.
4. Evaluating the outcome. The user checks the results of the computer execution.

The user forms a conceptual intention, reformulates it into the semantics of several commands, and eventually produces the action of keyboard input and studying the screen output by the computer.

Hansen's list of user engineering principles is a good starting point for developing a usage profile (3:523-532). The first principle is 'know the user'. This goal is simple

in concept but difficult to achieve. Most designers and developers assume that they know the users and their intentions. The reality of the situation is that other people learn, think, and solve problems in different ways than is anticipated. The ability of the user must not only be considered in the design of the software but also in the user manual, help screens, error messages, and tutorials.

Data entry is another area of consideration that is important for a well planned user interface. Data entry tasks can occupy a substantial fraction of the user's time. These tasks are the source of potentially dangerous errors. Smith and Mosier offer five high-level objectives for data entry (11:19). These objectives are the following:

1. Consistency of data entry transactions. The user should be required to perform the same sequences of actions.
2. Minimal input actions by user. If there are fewer input actions, then there is less effort required of the user and less chance for error. Selection from a list, such as a menu, reduces the need for memorization, structures the input process, and decreases the possibility of typographical errors. Redundant data entry should be avoided. The user should not have to enter the same information in two places because of the wasted effort and greater chance of making an error.

3. Minimal memory load on user. The design should reduce the need for the user to remember lengthy lists of codes and complex syntactic commands.
4. Compatibility of data entry with data display. The format of data entry should be closely related to the format of the displayed information.
5. Flexibility for user control of data entry. Experienced users may prefer to enter information in a sequence they can control.

These objectives are applied during the design process, and each project will need to develop these objectives into specific standards unique to its case.

Organizing the data display is another major area for consideration. Smith and Mosier offer five high-level objectives for data display (11:93). These objectives parallel those for the data entry and are as follows:

1. Consistency of data display. During the design process, the terminology, abbreviations, and formats should be standardized.
2. Efficient information assimilation by the user. The information format should be familiar to the user and related to the required tasks. This objective is met with neat columns of data, left justification for alphanumeric data, comprehensive labels, and proper spacing.

3. Minimal memory load on user. Arrange tasks so that the computer will perform the operation with few commands. This objective minimizes the chance of forgetting to perform a step.
4. Compatibility of data display with data entry. The format of displayed information should be clearly associated to the format of the data entry.
5. Flexibility for user control of data display. The user should get the information most convenient to perform the current task.

Again, as with the high-level objectives for the data entry, these high-level objectives are applied to the design process, and each project will need to amplify these into unique standards.

The wording of the messages and other information is important in the design of a system. Improvement of the design can be found in the layout of information on the screen. The information is presented to the user in the form of screen designs and error messages. Smith and Mosier present an extensive list of guidelines for information display (11) which is condensed to the following:

1. Display data to the user in directly usable form; do not make the user convert displayed data.
2. For any particular type of information display, maintain consistent format from one display to another.

3. Use short, simple sentences.
4. Use affirmative statements rather than negative statements.
5. Adopt some logical principle to order lists.
6. Ensure that labels are sufficiently close to be associated with their data fields but are separated from their data fields by at least on space.
7. Begin every display with a title or header, describing briefly the contents or purpose of the display; leave at least one blank line between the title and the body of the display.
8. For a large table that exceeds the capacity of one display frame, ensure that users can see column headings and row labels in all displayed sections of the table.

The error message should follow this general list of guidelines, but additional care should be given to phrasing and content of these messages since they can significantly impact user performance and satisfaction. Some considerations for error messages are the following:

1. Be precise as possible. Messages that are unspecific make it difficult to know what went wrong.
2. Be constructive and use a positive tone. The system should indicate what happened and what needs to be done. Avoid condemnation in error messages.

3. Choose user-centered phrasing. Wording that has the user in mind will suggest to the user that she is in control of the system.
4. Keep consistent visual format and placement. Most users prefer the upper- and lower-case messages for readability. Messages that are printed in only upper-case should be reserved for serious conditions.

Improved messages are of great benefit to the inexperienced user, but the regular user is also assisted.

Intentions of the Developer. The intentions of the system developer were consistent with the study of the literature. The system has the following characteristics: consistency, offers informative feedback, suggests simple error handling, and reduces short-term user-memory load.

1. Consistency is the reason for using identical terminology in prompts and menus. Using similar system commands also supports the consistency issue.
2. The information feedback given by the system to the user varies. For the minor system actions the response to the user is little or nothing, but for the major actions the response is more significant. For instructions, the user is the focus. Wordy messages are not used. 'Judging the user' is avoided, and a more constructive comment is given instead.

3. For simple error handling, the system will detect the error and give the user an appropriate message. For each command, all errors with the command are indicated to the user in the interest of efficiency. The system is designed so that the user can not cause a serious error because, if a flawed command is input, then the data state in the system is left unchanged. In providing the error protection for the user, there is a fine line to be followed between letting the user do something that he really wants to do while the operation may go against normal convention and the other side of the line where the system is dictatorial and possibly a source of frustration to the user. This fine line requires a balance between the automated control of the system and the human control of the user.
4. Miller states that the short-term memory capacity of the human is 'seven plus or minus two blocks' (7:795). This limitation in information processing required that the displays be kept simple and concise. Succinct organization of the screen display was the objective. Dense or messy displays can irritate, and inconsistent formats can confuse and slow down the user. The system offers an online menu for the command abbreviations and other pertinent information for error messages. The standard system

message is one line in length. An additional help facility was not achieved for lack of time. This additional help facility would print an explanation of greater length.

There is art in the design of an elegant user interface, and the intentions outlined here provided the basis for this design effort.

The phrasing of error and instruction messages seems to be more critical for influencing the user's perceptions. The wording impacts the performance and attitude of the user. Instructions are given for control of system use, and a good system will keep the user from feeling manipulated and poorly treated. The ultimate goal in the interface design was the following: as the user becomes engaged in system use, the system becomes a transparent tool so that they can concentrate on solving the problem or conducting research.

Equipment Used for Development

The Zenith Z-248 was used for system development. The Z-248 was much faster than the other computer systems that where available for use. The Z-248 had a color monitor with good resolution. The computer's speed and color screen provided less fatigue for the long hours spent in coding and validating the system.

System Validation

Validation of the developed system is based on its use in two U.S. Air Force Institute of Technology (AFIT) courses and in several decision analysis application theses. The system is used in OPER 7.82, Advanced Decision Analysis, and OPER 5.62, Introduction to Management Science. Three successfully completed theses include results from the developed system. The interaction with decision analysis thesis students provided a very valuable validation for the system.

Portability to Many Computers

Since the system was identified to be portable between a diverse set of computer systems and its development was primarily on an IBM PC compatible computer system, portability to various computer systems was investigated. System files were transferred to the AFIT Classroom Support Computer (CSC). These CSC files represented the central source for the transfer via modem to other computers. Along with the IBM PC compatible computer, the Amiga, Atari, and Macintosh computers also have a fully operational system.

User Manual Development

The user manual has many features incorporated for ease of use. The overall design of the manual is to present information that is easy to find, easy to understand, and all inclusive. System commands are presented in sections, and

each section is devoted to a stage of the influence diagram problem development and solution. Each command begins with a short description of entry with syntax and ordering of arguments. Examples of commands are also given. A complete sample session is included with comments.

The manual is to be included with the software code files, and this objective controlled what information was included in the manual. Since there is no capability yet to include graphics within the file code without requiring some additional software to make a graphics display, graphic representation of the influence diagram was left out of the manual. The objective of storage and distribution on diskette also restricted the amount of information that was included in the manual. There are sufficient examples for the user to perform the commands, but no representation of the types of error messages were included so that the memory space on the diskette was conserved.

The System Name

Without a doubt, one of the most trying tasks in developing the system was naming it. This section documents the more important considerations in a name and the reasons why 'PerForma' was chosen.

The name has to be interesting and draw the user to the program. The name is the user's initial view of the program, and should capture the essence of what the program does.

Consider what the system does: analyze an influence diagram. The influence diagram is a form of problem representation. The decision analyst is concerned with this form because it defines the problem in valuable terms. The computer software is concerned with the function of this form. The software system brings the analyst and the computer together by the means of form -- the influence diagram. The Latin equivalent of 'by the means of form' is per forma. Per meaning 'by means of; instrument by which anything is done,' and forma meaning 'form, shape, beauty.'

There are three elements captured in the name PerForma. The first element is a Latin sense implying classical roots. The second element is an English derivative with good meaning lacking negative connotation. This element will draw on the familiarity the user has with modern usage: words like perform and performance. The third element is the combination of the Latin words into one word giving these classic words an update to modern times.

Summary

PerForma was developed meeting the objective to have a software system for analyzing and solving influence diagrams that is widely and easily accessible to the decision analysis community at AFIT, and this chapter discussed the methodology involved in meeting this objective. XLISP was used to write the system. This language met the system requirements for accessibility on a wide variety of computer systems. The

user interface development followed the theory of friendly software design. The system validation is based on its use in two AFIT courses and in numerous application theses. The investigation of computer portability found the Amiga, Atari, Macintosh, IBM PC and PC compatible computers to have a fully operational system. The system manual incorporates several features for the user's convenience including a documented session of system use. PerForma also has a tutorial which includes a definition of the influence diagram and an explanation of modeling an influence diagram in the system while analyzing it in the decision analysis cycle. The tutorial is presented in the next chapter.

IV. PerForma Tutorial

Introduction

The influence diagram is a form for modeling decisions and associated uncertain variables while precisely revealing the relationship between them. The influence diagram is the framework that ties the information flow and the variable dependence together into a graphical structure. The influence diagram is hierarchical, containing a top level graph with a second level as data, and the computer readily manipulates this graphical structure.

The tutorial has several purposes. First, the influence diagram is introduced. Second, the influence diagram is modeled with the computer software system PerForma. Third, the basic operations for influence diagram transformation are presented. Finally, the influence diagram evaluation procedure is given.

The Influence Diagram

The influence diagram is a directed network of nodes. The nodes represent decisions, random variables, and values. The directed arc into the decision represents information that is available when the decision is made, and, therefore, there is an assumption of time precedence for the decision node. An arc into the random variable node represents probabilistic dependence. All probability distributions in

the influence diagram together specify a joint probability distribution.

The influence diagram can contain three types of nodes: decision nodes, chance nodes, and deterministic nodes. The graphical representation of each node is different to distinguish between the types. The decision node is represented as a square. The chance node is represented by a circle. The deterministic node is designated by a double circle. The value node, a specific type of deterministic node, is designated by a diamond. Figure 1 illustrates the types of node symbols.

The arcs of the influence diagram also have important distinctions. The directed arcs in the graph are of two types: informational and conditional. The informational arcs are into the decisions, and the conditional arcs are into the chance, deterministic, and value nodes.

The decision node represents a choice among alternatives. The alternatives form a set, and the alternative chosen from this set maximizes the problem's expected value represented by the value node. An arc into the decision node indicates that the information at the source of the arc is available at the time the decision is made. Figure 2a shows an example of an informational arc. The arc indicates that the information from chance node x is available when decision d is made.

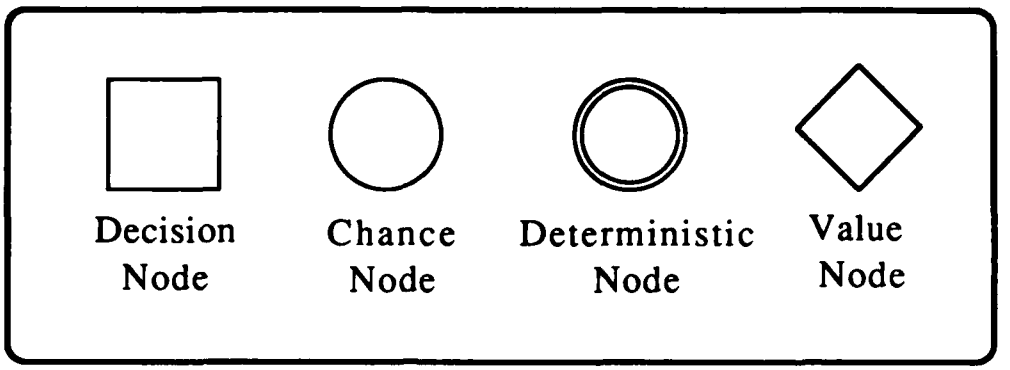


Figure 1. Types of Nodes.

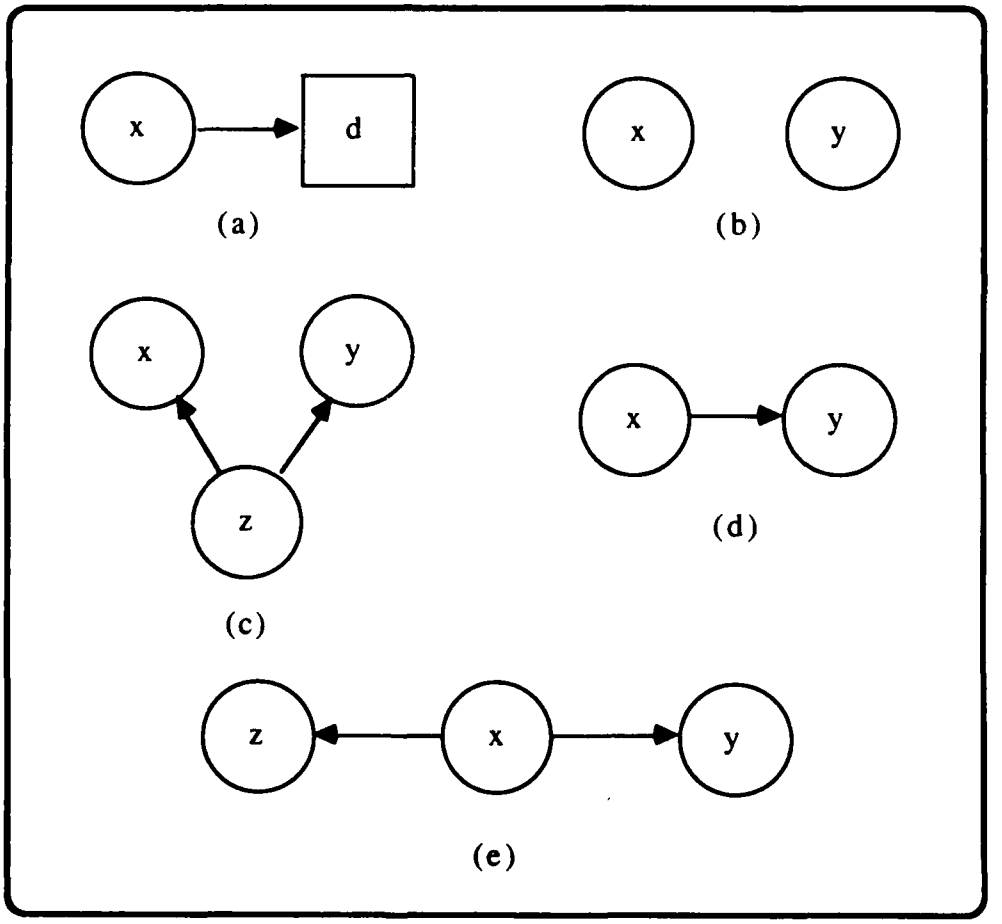


Figure 2. Implications of Directed Arcs.

The chance node represents a random variable with a probability distribution. The type of probability distribution depends upon the random variable's conditional dependence or the lack thereof. If there is no path between two chance nodes, then they are independent. Figure 2b displays an example where chance node x is independent of chance node y. If there is no direct arc between the two chance nodes but an undirected path exists, then they are conditionally independent. As illustrated in Figure 2c, the independence of chance nodes x and y is conditioned on the chance node z. If there is an arc between the chance nodes, then the probability distribution for the node at the end of the arc is conditioned on the node at the arc's origin. Referring to Figure 2d, the conditional probability distribution for chance node y is conditioned on chance node x. If a chance node has no arcs into it , then the node contains an unconditional (marginal) probability distribution. Node x in both Figures 2d and 2e contains an unconditional probability distribution.

The deterministic node can exist in two possible states: as a regular deterministic node and as a value node. The value node represents the decision maker's value to be maximized in expectation. At most one value node can exist for the influence diagram, and it is placed at the end of the directed network. The influence diagram is oriented if it contains a value node.

The arc into a node denotes precedence. The node at the head of the arc is known as the successor to the node at the arc's origin. The node at the arc's origin is the predecessor to the node at the arc's termination. The arc between two chance nodes can be reversed by invoking Bayes' Theorem (a mathematical formula that reverses probabilistic conditioning), and, in effect, the conditioning (precedence) is reversed. The arc into or out of a decision node cannot be reversed because the reversal would change the meaning of the influence diagram.

No cycles can exist in the influence diagram. If a cycle should contain decision nodes, then the cycle would contradict the assumption of time precedence. In other words, a cycle of decisions implies that the information for a decision is not fully known at the time the decision is made. If a cycle contained a decision and a random variable, then it would imply the decision maker can know something about a decision he has not made yet. A cycle of random variables makes the computation of the underlying joint distribution impossible.

Figure 3 exhibits some different examples of influence diagram models. In model (a), the value node depends on the random variable or chance node, and the random variable is conditioned on the decision. The chance node will contain a probability distribution conditioned on the decision node. The decision node is the predecessor of the chance node.

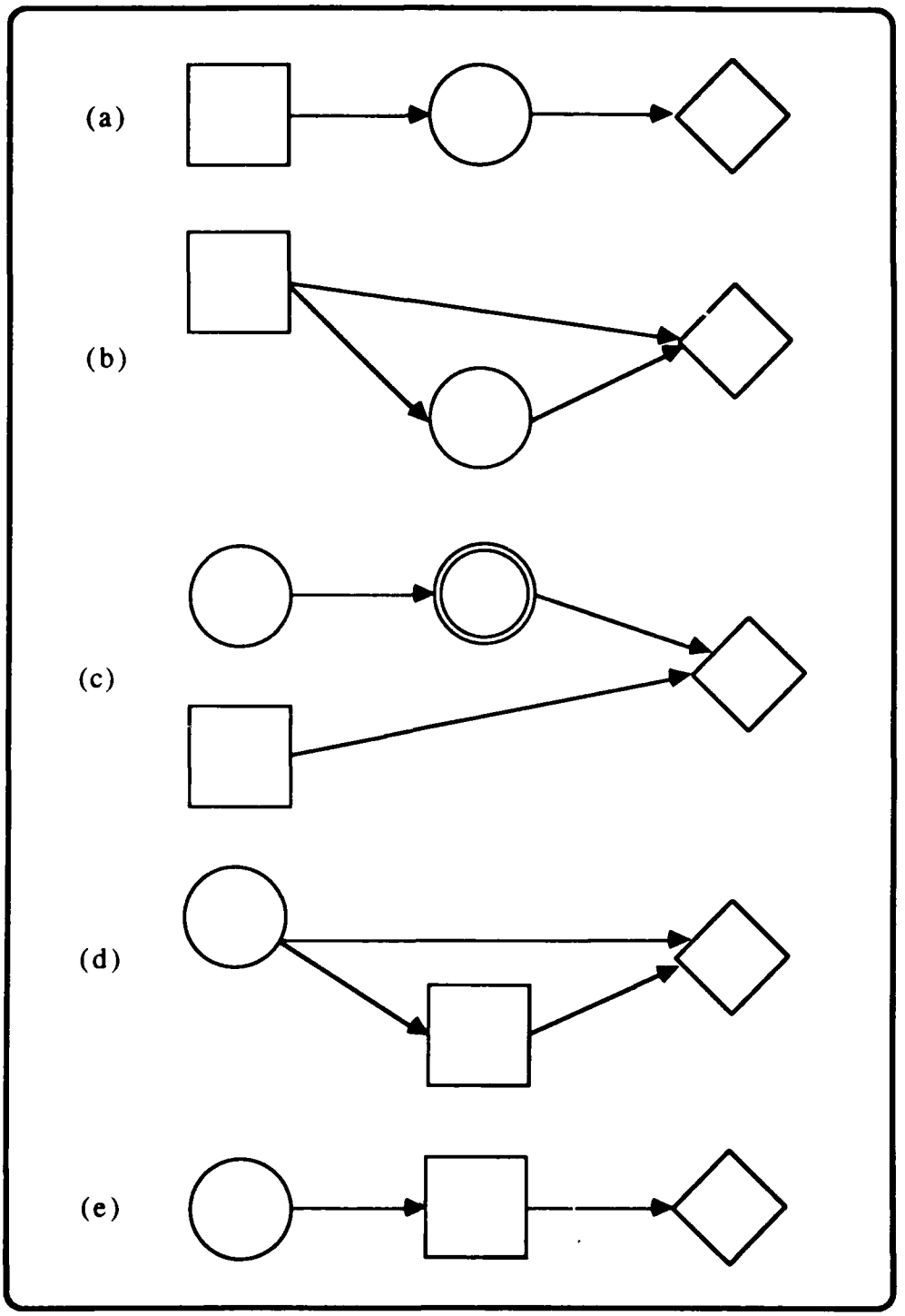


Figure 3. Examples of Influence Diagrams.

The successor of the chance node is the value node. The influence diagram in model (b) is similar to model (a), but there is an additional arc from the decision node which leads to the value node. This arc implies that the value in model (b) is dependent on both the random variable and the decision. Note that there is no cycle in this influence diagram since the directed arcs never create a looped path. For model (c), the value depends on the deterministic node and the decision node. The deterministic node is conditioned on the chance node. However, the decision and chance nodes are independent. Model (d) is another case where the value depends upon the decision and random variable, but the outcome of the random variable is known before the decision is made. The informational arc into the decision node implies this information is available when the decision is made. In model (e), the outcome of the random variable is known at the time the decision is made, but the random variable does not influence the value. That is to say, there is no arc from the chance node to the value node. Therefore, the chance node is unnecessary to the influence diagram and has no effect on the value derived from the decision. This model gives a good example of how the influence diagram explicitly reveals the dependency and information flow for important and relevant consideration. This chance node can be disregarded because the information it provides is irrelevant.

According to Shachter, the influence diagram is considered to be regular when the following conditions are satisfied:

1. the directed graph has no cycles,
2. the value node, if present, has no successors, and
3. there is a directed path which contains all of the decision nodes (10:875).

The last condition is the same as requiring a sequential ordering of all decisions.

The sequential ordering of all decisions implies that any relevant information available at the time of one decision should be available for all subsequent decisions. This property is known as the 'no forgetting' property. If a decision node precedes another decision node in a regular influence diagram, then the first decision node and its direct predecessors should directly precede the second decision node. This precedence will be denoted by the addition of arcs to the second decision node during the application of the evaluation procedure presented later. If the 'no forgetting' arcs are added before the evaluation begins, then the additional arcs clutter the influence diagram. Figure 4a illustrates the influence diagram in the initial form. Figure 4b (to be discussed below) illustrates the influence diagram with 'no forgetting' arcs accounted for.

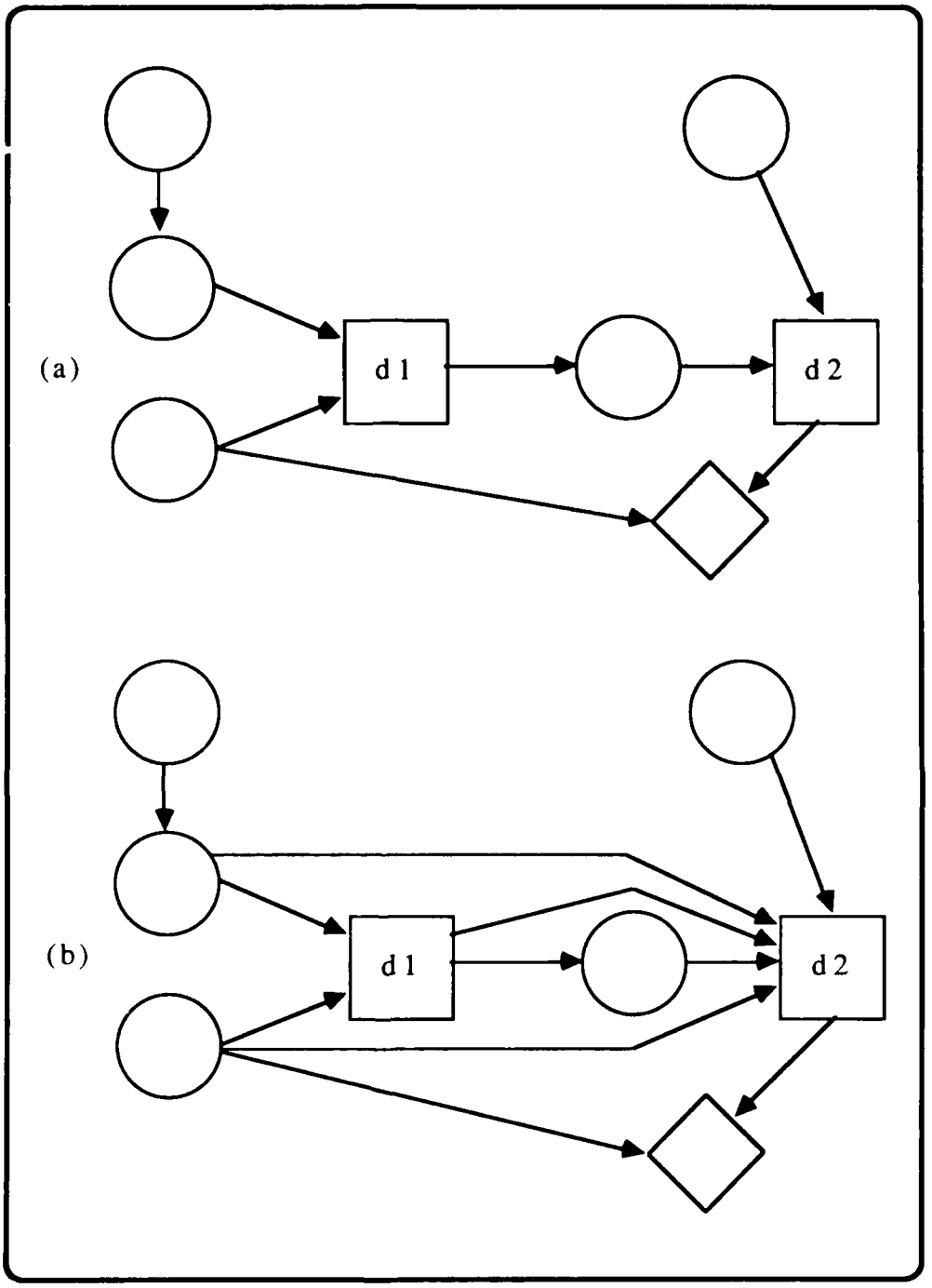


Figure 4. Addition of "No forgetting" Arcs.

Modeling the Influence Diagram with PerForma

Once the influence diagram is formulated, the diagram is modeled in the PerForma computer system for solution and further analysis. The commands used are described in the PerForma documentation.

The first step in modeling an influence diagram is to create a new influence diagram with the NEW-ID command. The user must designate the name of the new influence diagram, and all subsequent commands will be performed on this diagram. The SHOW-ID command will display the entire influence diagram.

The next step in modeling the influence diagram is to create nodes. This step can be started with either the SET-NODES or ADD-NODE command. The SET-NODES command allows many nodes to be added at once, and the ADD-NODE command allows one node addition at a time. The node kind will be 'decision', 'chance', 'det' (denoting deterministic), or 'value.' The decision node will have the alternatives set during the SET-NODES operation. Otherwise, the alternatives are input with either SET-ALTS or ADD-ALT command. The chance node will have its outcomes set during the SET-NODES operation. Otherwise, the outcomes are input with either the SET-OUTCOMES or ADD-OUTCOME command. The SHOW-NODE command will show the information for the node.

The predecessors of the nodes should be set next with either the SET-PREDS or ADD-PRED command. Since the value

node must be placed at the end of the influence diagram, the value node can not be a predecessor to any other node. PerForma will not allow chance and decision nodes to have deterministic node predecessors. Once the predecessors for all nodes are set in PerForma, the arcs for the influence diagram are defined.

The graph structure consists of all nodes in symbols and their directed arcs. The arc information translates to the predecessors and successors of each node. The SHOW-GRAPH command will show the graph structure of the influence diagram in word format. The printed information gives the node name, kind, predecessors, and successors for all nodes of the diagram. PerForma does not have graphics capability, so the graph structure information must be drawn by hand to depict the true graph structure in node symbols and arcs.

The chance node can be further developed by defining the probability distribution. The probability mass function (discrete probability distribution) will be loaded with the SET-PROBS command. If the chance node has any predecessors, the predecessor(s) must have their outcomes or alternatives specified. The conditional dependence implies the probability distribution will be a conditional probability mass function. The SHOW-PROBS command will show the probabilities for a chance node.

The deterministic node can have its value(s) assigned after it has been created, but if the node has any

predecessors, the predecessor(s) must have their outcomes or alternatives specified. For a value node, the individual values correspond to the values at the end-points of a decision tree. The values can be input by one of two methods: by input of individual values or by setting a function and then creating the values from the function. The SET-VALS command is used to input individual values into the node. The SET-FUNCTION command in conjunction with the BUILD-VALS command is used when a deterministic function defines the values. The SET-FUNCTION command is used first to input the function, and then the BUILD-VALS command transforms the function into the individual values. If a function is used to define the values and the deterministic node has conditional dependence, then the outcomes for each predecessor must be numbers (this restriction can be relaxed with LISP programming). These numbers will be used in the function to create the values. Since the value node is a type of deterministic node, the same procedure of assigning values applies to a value node as well. The SHOW-VALS command shows the values for a value or deterministic node.

Editing of probabilities and values for nodes is available. Changing the chance node's probabilities or the deterministic node's values is done with either the CHANGE-PROBS or CHANGE-VALS command, respectively.

Since PerForma does not automatically handle the 'no forgetting' property of the influence diagram, the addition

of the 'no forgetting' arcs is done by user with either the SET-PREDS or ADD-PRED command during the solution phase. Figure 4b illustrates where 'no forgetting' arcs are present in a sample formulation.

Analysis of the Influence Diagram

The analysis of the influence diagram consists of a sequence of transformations. These transformations neither change the maximum expected value of the influence diagram nor alter the optimal policy to achieve that maximum expected value. Since the transformations have these properties, they are known as value-preserving. A node is removed when it is processed by a value-preserving transformation and its information is no longer needed. The value-preserving transformations are applied to the nodes toward the goal of reducing the influence diagram to a single value node. The transformations consist of barren node removal, chance node removal, decision node removal, and arc reversal.

Any node besides a value node that has no successors is called a barren node or widow node. The barren node could be affected by other nodes, but it affects no other node. Therefore, the barren node may be removed from the influence diagram. If the barren node is a chance node, then its outcome does not affect any other node in the diagram. If the barren node is a decision node, then any alternative would be optimal since the decision affects nothing. The

barren node removal for the influence diagram in PerForma is done with the DELETE-NODE command. The transformation of barren node removal is illustrated in Figure 5 where the influence diagram modeled in (a) is transformed into the influence diagram in (b).

Like the barren node removal, the chance node removal is a transformation with one condition. If the value node is the only successor of the chance node, then the chance node may be removed by expectation. After the calculation of the expected values, the chance node is no longer in the influence diagram, and the value node inherits the chance node's direct predecessors as its direct predecessors. The chance node removal is also known as 'taking the expectation of the value node with respect to the chance node.' The PerForma command that performs the chance node removal is the EXPEC command. Figure 6 shows the chance node removal. The transformation is made to the influence diagram in (a) to create the influence diagram in (b).

Before the transformation of decision node removal can be made several conditions must be satisfied. One condition is that the value node must be the only successor of the decision node. The other condition to be satisfied is that all direct predecessors of the value node (except the decision node) are direct predecessors of the decision node. The decision node is removed by maximizing the expected value, and this transformation is also known as 'maximizing

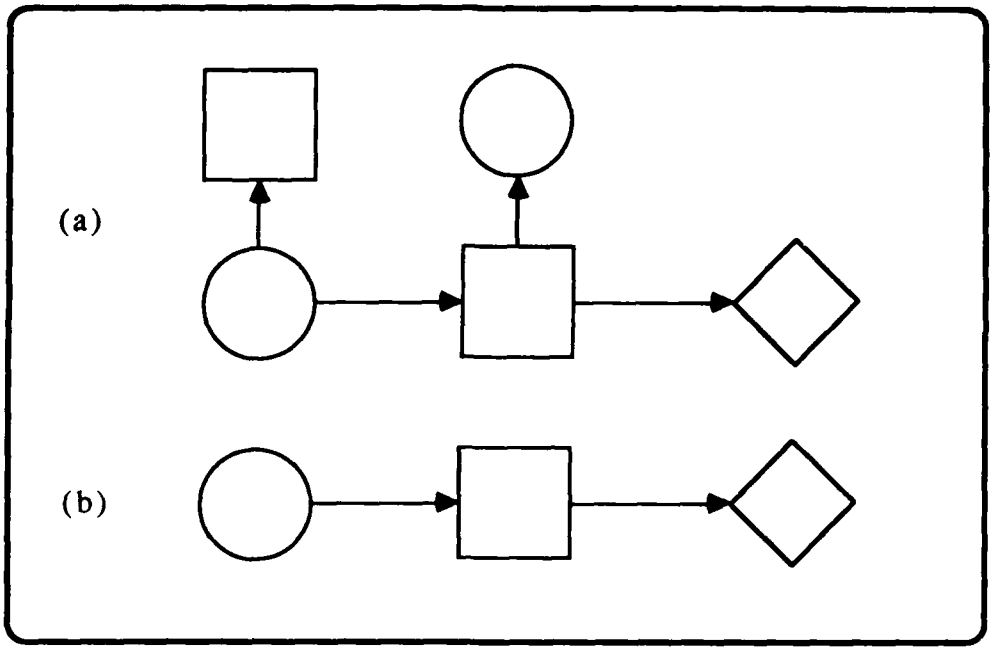


Figure 5. Barren Node Removal.

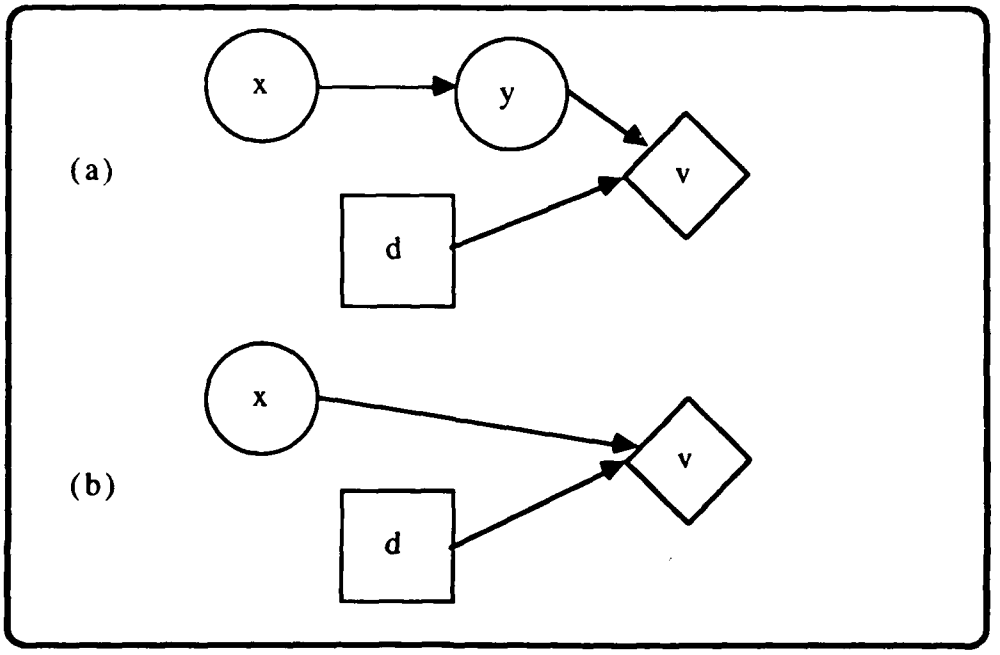


Figure 6. Chance Node Removal.

the value node with respect to the decision node.' The value node does not inherit new conditional predecessors as a result of this transformation, and, therefore, some of the informational predecessors of the decision node could become barren nodes. The PerForma command that accomplishes this transformation is the MAXIM command. The maximizing alternative(s) are recorded as the optimal policy, and the PerForma command to show this policy for the decision node is SHOW-D*. PerForma will keep in the system the information for the maximized decision node and denote the node kind as decn-det. Figure 7 shows the decision node removal from the influence diagram (a) to become (b).

The arc reversal transformation can be applied to the arc between chance nodes but, as noted above, an arc involving a decision node cannot be reversed. The condition for this transformation is straightforward: there is no other path between the two nodes. That is to say that there is no other path from the node at the arc's origin to the node at the arc's termination except for the one arc itself. This condition must be satisfied because if an alternate path existed, then reversing the arc would create a cycle in the influence diagram. The arc is reversed using Bayes' Theorem. After the arc reversal, both chance nodes share the same direct predecessors because the two nodes now share the same information. The PerForma command that does the arc reversal is the BAYES command. Figure 8 shows the

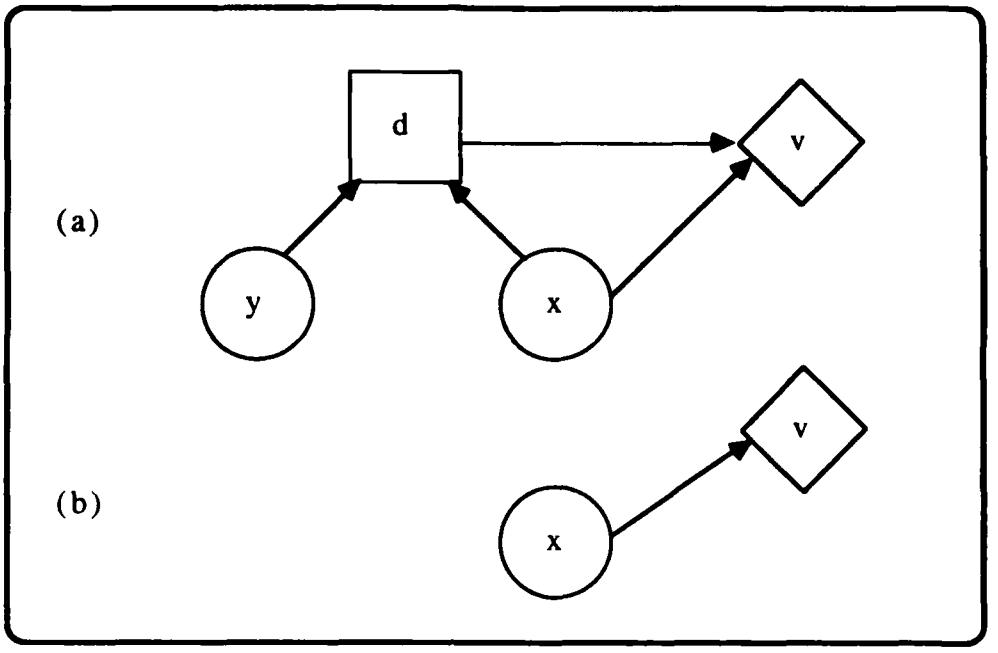


Figure 7. Decision Node Removal.

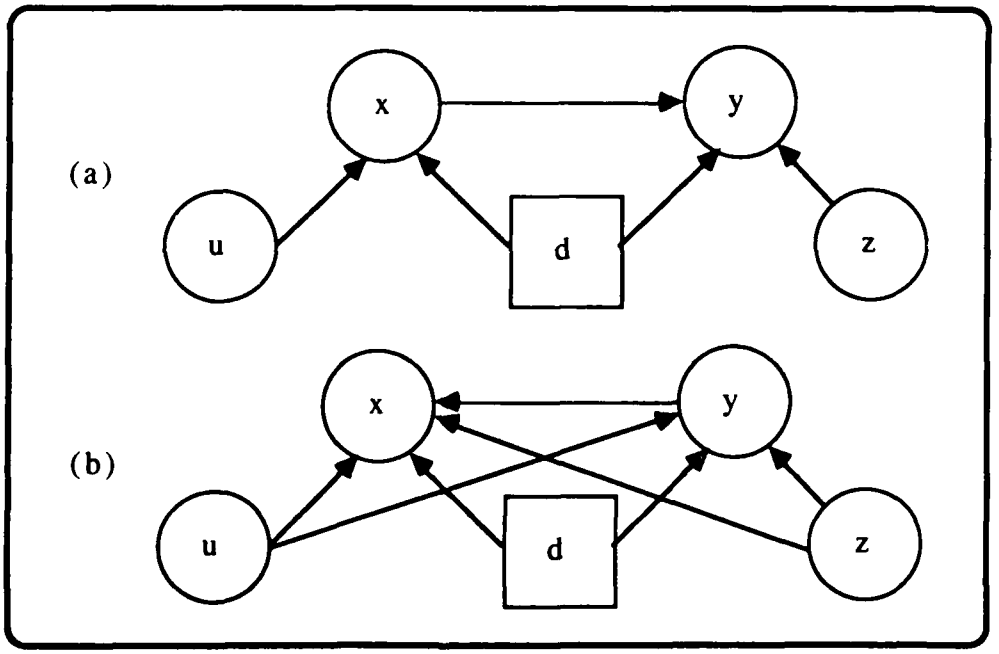


Figure 8. Chance Node Arc Reversal.

graphical representation of the arc reversal between node x and node y with its result in (b).

Influence Diagram Evaluation Procedure

The procedure for evaluating influence diagrams takes the transformations that have been presented and manipulates the diagram to produce a solution (10:879). The solution is achieved when the nodes have been removed until only the value node remains. When the value node has no predecessors, the value node contains the maximum expected value for the influence diagram, and the optimal alternative is stored in the former decision nodes.

The algorithm is as follows:

```
BEGIN
    check influence diagram for the following: 1. no
        cycles, and 2. if it contains a value node, then
        the value node must have no successors
    add 'no forgetting' arcs
    delete all barren nodes
    remove all deterministic nodes
    WHILE value node has predecessors, do the following:
        IF there is a chance node that has only the value
            node as its successor, then remove it.
        ELSE IF there is a decision node that has only the
            value node as its successor, and all other
            direct predecessors of the value node are
            direct predecessors of the decision node,
```

then remove the decision node and delete all barren nodes that result.

ELSE there must be a chance node that has as successors the value node and chance node(s). Reverse the arc to the other chance node(s) until the value node is the only successor, and remove the chance node. There are conditions under which the arc reversals may create a cycle, so the arc reversals must be ordered.

END WHILE

END

This algorithm is the influence diagram solution procedure, and given a value node, any regular influence diagram can be evaluated to a unique solution (10:879).

If no decisions are involved in the model, then the information presented by the influence diagram is the relationships between the random variables. Probabilistic inference can be performed on the model using PerForma.

Conclusion

The algorithm enhances the exploitation of the influence diagram. The model formulated as an influence diagram can be evaluated in the same form. The decision makers and experts that are involved with the formulation can now better understand the process of evaluation. This

understanding hopefully brings greater confidence in the values and optimal policies presented by the solution.

V. Areas for Further Development and Conclusions

Areas for Further Development

Based on the observations made during the software development, the following recommendations are proposed for further development of PerForma.

The development of a graphical display of the influence diagram is the most useful. A graphical display would use the full potential of the influence diagram as a communication form between the analyst and the decision maker. As the system stands, a clear textual representation of the graph structure is provided, but the graphical display of the influence diagram must be managed by the user. During the formulation and analysis, the user must draw by hand the influence diagram and update the drawing when a mathematical operation is performed.

A solution algorithm should be incorporated into the reduction commands for a thorough influence diagram analysis. Once the user has built his influence diagram into PerForma, then a single command to solve the diagram should be available. There are many considerations for implementing the algorithm. One of the most important considerations would be the order of node removal. The ordering of mathematical operations affects the memory requirement, and the number of mathematical operations affects the length of time to solve the influence diagram.

The applications for PerForma would be greatly expanded if continuous decision and random variables could be accommodated. The work of Kenley makes this implementation possible (6).

Influence diagrams can model dynamic programming problems. The dynamic programming operations and algorithm should be incorporated into PerForma along with checking an operation's validity and careful creation of user displays. The work of Tatman makes this system development possible (13).

PerForma should contain features that automate stages of the decision analysis process. These enhancements could include automatic worth lotteries, automatic stochastic sensitivity analysis, and the ability to interface with other software. The user could have the worth lottery of the value function automatically calculated. The user could perform automatic sensitivity analysis of the value function relative to a specific node. The interface with other software could first be explored with a spreadsheet since a spreadsheet is effective for building the value model and performing sensitivity on the model.

Finally, a help facility would enhance the user interface. One suggestion for the help facility is to have a command that prints an explanation to the screen of a particular command the user chooses.

Conclusions

A software system for influence diagram analysis, PerForma, now exists that is accessible to the students at the U.S. Air Force Institute of Technology (AFIT) and the U.S. Air Force. Many analysts are currently using this system for academic education and in strategic decision analysis applications.

The system makes an extremely powerful modeling framework (for both formulation and analysis) easily accessible to the AFIT community. PerForma runs on a broad spectrum of personal computers. The system is cost free and has no distribution restrictions except for commercial use. PerForma is already a foundation for a 700 level course in the Operational Sciences Department at AFIT. The system will be essentially handed out to every student on the first day of class in MA 5.70 Decision Analysis.

PerForma is now making a significant contribution to the Air Force's ability to analyze strategic decisions and is affecting other agencies of the DOD as well. The system was used in the analysis incorporated into several successful decision analysis application theses and those results passed to the sponsor and other interested organizations. These organizations include the 549th Weapon System Evaluation Squadron at HQ SAC, Directorate of Propulsion, Program Control at HQ ASD, Studies and Analysis

at HQ ATC, and U.S. Army Armament Research, Development and Engineering Center.

PerForma is a professional product that will continue to make a significant contribution in the research and analysis of the strategic decision.

Appendix A:

User's Manual with Documented Session of System Use

The PerForma user's manual includes an introduction, a note from the authors, general notes, user commands, and a documented example of system use. The user commands are divided into functional sections: system commands, build commands, reduction commands, and show commands. The example of system use is labeled Session1. Session1 solves a decision analysis problem on a small scale, but it requires all of the fundamental influence diagram transformations while applying the influence diagram evaluation algorithm.

The PerForma documentation is included with the other system files for ease of distribution. This documentation is a computer file named PerForma.DOC. The file can be read in its present form typed to the computer screen or printed with a computer printer without reformatting. The spacing for screen display and page breaks for print out are included in the computer code for the user's convenience. The documentation is brief and assumes a basic knowledge of influence diagrams and the concepts for their solution.

INTRODUCTION

PerForma is a computer software system that enables you to analyze influence diagrams in the framework of decision analysis. The system has been designed to be highly portable, and it currently is operating on the Amiga, Atari, MacIntosh, IBM-PC and PC compatible computer systems.

PerForma is written in XLISP, a conservative dialect of LISP and requires XLISP Version 1.7 to operate.

This document is a brief description of PerForma and assumes the user has a basic knowledge of influence diagrams and the concepts for their solution. A thorough introduction to influence diagrams is the article 'Evaluating Influence Diagrams', by Ross D. Shachter, published in Operations Research, Volume No. 34 (November-December 1986).

A NOTE FROM THE AUTHORS

If you have any problems with PerForma, please feel free to contact Dr. Tatman at the address below for help. Each user has particular needs. If you think of ways in which this system can be extended and enhanced to make your analyses better, please write. Any suggestions for improvement will be welcomed.

Dr. Joseph A. Tatman
AFIT/ENC
Wright-Patterson AFB, Ohio 45433

Influence diagrams are opening the science of decision analysis to great possibilities. The influence diagram is a form for communicating the problem structure not only between the decision-maker and the decision analyst but also between the decision analyst and the computer. Hence, we have named this system from the Latin "per forma", meaning "by means of form". We hope you find PerForma useful in the exciting development of decision analysis.

GENERAL NOTES

All commands work on a single influence diagram. This influence diagram is the last one created by the command 'NEW-ID' or the last diagram that was loaded using the command 'LOAD-ID'.

A single apostrophe before a symbol (e.g. 'x) tells the XLISP interpreter to pass along x directly as your input. You will normally have to put the single apostrophe in front of all your
node names, list of outcomes, etc. when you provide them as input to commands.

The names used in the system can consist of any sequence of non-blank printable characters except the following:

( ) ' ` , ' ;

Since the symbol ' cannot be used in names, the normal convention in decision analysis of quoting the name of the variable, such as 'weather' to represent a test on the weather, cannot be used. Another convention will have to be applied, and we leave it to you to find one that you feel most comfortable with. For example, the node name TEST-WEATHER could be used.

Uppercase and lowercase characters are not distinquished within
the system. All lowercase characters are mapped to uppercase on input.

The commands are fairly simple, but attention must be paid to the syntax and content of each command. When referring to the examples of command input, everything between the double-quotes including the parentheses is intended as the system entry. The commands must be correct with the placement of parentheses, order of node names, etc. If an entered command has an error and an error message is printed, then
enter <Ctrl-G>, or '(top-level)' to return to the normal system prompt and re-enter the command.
Several commands will give a warning of the operation's result and ask the user if they want that result and wish to continue with the operation. The user must enter a 'Y' for a yes answer or a 'N' for a no answer. If another answer besides a Y or a N is given, then the system will restate the warning and question. This feature is a protection against destroying valuable work.

PerForma performs checking for ensuring the commands for the operations and reductions are performed in the correct order and that they are indeed valid.

To solve an influence diagram you have to reduce the influence diagram one operation at a time. There is no solve command to automatically solve the influence diagram.

To use the system, XLISP17.EXE must be present in the directory in which the PerForma files exist. To start the system, just type 'XLISP17' while in the directory that contains XLISP17.EXE and the PerForma files. PerForma will automatically be loaded. To exit the system, type '(exit)' at the system prompt.

PerForma provides the option of creating a transcript file of your work session. A transcript file is a record of all screen input and output. The first input asked by the system is the designation of this transcript file. If you do not want a record file of the current session, then enter NIL. If you want a transcript, then enter a transcript filename that you wish to use. The transcript filename can be up to eight characters in length consisting of any alphanumeric characters with no spaces included. File extensions are permitted with the normal convention of three characters. The transcript file will be stored in the default directory. The transcript file can contain comments which are helpful in reconstructing the sequence of events. The system will ignore semicolons and anything that appears after them on the same line. To insert a comment, type your comment after placing a semicolon anywhere on a blank line or after the system command that you want performed.

The section of this document labeled SESSION1 is a file created with the transcript option, and it contains an example of system use.

SYSTEM COMMANDS

System commands are not associated with a particul stage of influence diagram manipulation. They are available at any time the user wishes.

EXIT

Enter: (EXIT)
Terminates the XLISP/PerForma session and return you to your computer's operating system.

LOAD-ID

Enter: (LOAD-ID 'file-name-to-be-loaded)
Loads an influence diagram previously stored in file into the system. The command '(LOAD-ID 'george)' will load the influence diagram stored in the default directory under the name 'george.lsp'. If there is an influence diagram already in the system, then the user will be given a warning and asked if he wishes to erase the present influence diagram and load one from the file. The loaded influence diagram will become the current influence diagram and all subsequent commands will be performed on it. This influence diagram will exist until the next LOAD-ID or NEW-ID command is performed.

MENU

Enter: (MENU)
Gives a menu of the PerForma system commands and related LISP commands. This menu is brief in the information given. If you need more particulars, then check this document.

STORE-ID

Enter: (STORE-ID 'filename-to-store-current-id)
Stores the current influence diagram to the filename given. The command '(STORE-ID 'george)' will store the current influence diagram under the name 'george.lsp' and in the default directory. You may choose an entirely different name than the name of the current influence diagram. The name may be up to eight characters in length and consist of any alphanumeric characters. Do not put any file extension (i.e., .lsp) on the filename because the system will add the '.lsp' extension. If the mistake of putting a file extension on the filename is made, then error messages will be printed to the screen. To get back to a normal system prompt, enter '(top-level)'. Caution must be exercised in choosing the name for the storage file. If you have an existing file in the default directory which has a filename that you choose to use (with the .lsp extension), then the existing file will be overwritten and lost.

BUILD COMMANDS

The build commands are used for creating and editing an influence diagram. The validity of the operation called for will be dependent on the information that already exists for the influence diagram. For example, the values for a value node can not be set without the predecessors of the value node being fully defined (i.e., having outcomes or alternatives).

ADD-ALT
Enter: (ADD-ALT 'alternative-name 'decision-node-name)
Adds a new alternative to a decision node which might already have a set of alternatives. For example, the command '(ADD-ALT 'h 'd)' will add the alternative h to the list of alternatives that decision node d might already have.

ADD-NODE
Enter: (ADD-NODE 'node-name 'node-kind)
Adds a node to the present influence diagram. The command consists of the node name and the kind of node. There are four possible node kinds: chance, decision, deterministic (denoted in the system as det), and value. The command '(ADD-NODE 'x 'chance)' adds a chance node x to the current influence diagram. The command '(ADD-NODE 'd 'decision)' adds a decision node d to the influence diagram. The command '(ADD-NODE 'u 'det)' adds a deterministic node u to the influence diagram. The command '(ADD-NODE 'v 'value)' adds a value node v. The user may also use the command SET-NODES to add several nodes to the influence diagram at once.

ADD-OUTCOME
Enter: (ADD-OUTCOME 'outcome-name 'chance-node-name)
Adds a new outcome to a chance node which might already have a set of outcomes. The command '(ADD-OUTCOME 'g 'x)' will add the outcome g to the list of outcomes that chance node x might already have. The command SET-OUTCOMES can also be used to add an outcome.

ADD-PRED
Enter: (ADD-PRED 'node-name-of-predecessor 'node-name)
Adds a new predecessor to the current list of predecessors of a node. The command '(ADD-PRED 'y 'x)' adds node y to the current list of predecessors of node x. The command SET-PREDS can also be used to add a predecessor.

BUILD-VALS
Enter: (BUILD-VALS 'value-or-det-node-name)
Builds values from a function given for a value or det node. If a function defines the values for a deterministic or value node (i.e. SET-FUNCTION command used to define values), then the BUILD-VALS command takes the function and builds an array of its values. Entering the command '(BUILD-VALS 'v)' will build the values for node v. If the node is a value node, these values correspond to the values that would appear at each endpoint of a decision or probability tree. The operation could be time consuming with a large number of combinations of the predecessors' outcomes and alternatives. It will not be uncommon to see the process take upwards of an half-hour to an hour.

CHANGE-PROBS
Enter: (CHANGE-PROBS 'chance-node-name)
Changes the probabilities for a chance node. The command '(CHANGE-PROBS 'x)' can change the outcome probabilities that chance node x already has. If one or more predecessors of the chance node exist, this operation will require user input at each probability vector. The user must input a 'Y' to change the printed probabilities for that line. Otherwise, the user will enter a 'N' to not change the probability vector. To change the vector, enter 'Y' and the system will prompt for the new list of probabilities (e.g., '(.3 .2 .5)'). After changing the vector, the system will ask if more probabilities for the node are to be changed. The command SET-PROBS can also be used to set or change probabilities. If you have a large number of probability vectors and only wish to change a few vectors, then the change-probs command is recommended.

CHANGE-VALS
Enter: (CHANGE-VALS 'value-or-det-node-name)
Changes the values for a value or det node. The command '(CHANGE-VALS 'v)' can change the values that node v already has. Node v must be either a value or a determistic node. If one or more predecessors of node v exist, this operation will require user input at each value. The user must input a 'Y' to change the printed value for that line. Otherwise, the user will enter a 'N' to not change the value. To change the value, enter 'Y' and the system will prompt for the new value as a single number. After changing the value, the system will ask if more values for the node are to be changed. The command SET-VALS can also be used to set or change values. If you have a large number of values and only wish to change a few, then the change-vals command is recommended.

DELETE-NODE
Enter: (DELETE-NODE 'node-name)
Deletes a node from the influence diagram. The command '(DELETE-NODE 'x)' deletes node x. This command should only be used if x is a barren node or if you are editing and not solving the influence diagram.

DELETE-NODES
Enter: (DELETE-NODES '(list of node names))
Deletes all nodes given in a list from the influence diagram. The command '(DELETE-NODES '(x y z))' deletes nodes x, y, and z. The nodes must be listed in parentheses. This command should only be used if x, y, and z are barren nodes or if you are editing and not solving the influence diagram.

NEW-ID
Enter: (NEW-ID 'new-empty-id-name)
Creates a new, empty influence diagram and assigns it the name given. The command '(NEW-ID 'george)' will create a new influence diagram named george. If there is an influence diagram already in the system, then the user will be given a warning and asked if he wishes to erase the present influence diagram and create a new one. The newly created influence diagram will become the current influence diagram, and all subsequent commands will be performed on it. This influence diagram will exist until the next NEW-ID or LOAD-ID command is performed.

SET-ALTS
Enter: (SET-ALTS 'decision-node-name '(list of alternatives))
Sets the alternatives of a decision node in the current influence diagram. The command '(SET-ALTS 'd '(a b c))' will set the alternatives of the decision node d to a, b, and c. The alternatives must be listed in parentheses.

SET-FUNCTION
Enter: (SET-FUNCTION 'value-or-det-node-name '(function))
Sets the deterministic function of a value or deterministic node. An example use would be '(SET-FUNCTION 'v '(+ x y z))' where v is any deterministic or value node and nodes x, y, and z are predecessors of v. The function must be written in the LISP programming language syntax. An example function written in FORTRAN is given as the following:

```
(x**2) + 3*exp(y) + sin(sqrt(z)).
```

The equivalent LISP code would be as follows:

```
(+ (expt x 2) (* 3 (exp y)) (sin (sqrt z))).
```

A recommended text for LISP reference is LISP, Second Edition, by Winston and Horn. The documentation XLISP17.DOC has particular information for XLISP Version 1.7.

SET-NODES
Enter: (SET-NODES)
Sets several nodes into the influence diagram at once. The command will prompt for the node name, and kind of node. If the node kind is decision or chance, the system will also prompt for a list of alternatives or outcomes, respectively. One node can be set into the influence diagram as well.

SET-OUTCOMES
Enter: (SET-OUTCOMES 'chance-node-name '(list of outcomes))
Sets the outcomes of a chance node that is in the influence diagram. The command '(SET-OUTCOMES 'x '(a b c))' will set the outcomes of chance node x to a, b, and c. The outcomes must be listed in parentheses.

SET-PREDS
Enter: (SET-PREDS 'node-name '(list of predecessors))
Sets the list of predecessors for a node that is present in the influence diagram. The command '(SET-PREDS 'd '(x y z))' sets the predecessors of node d to nodes x, y, and z. The predecessors must be listed in parentheses.

SET-PROBS
Enter: (SET-PROBS 'chance-node-name)
Sets the probabilities for the various outcomes of a chance node. This command will query the user to provide the probabilities for the chance node. You must have all predecessors fully defined including the outcomes or alternatives for each node. An example of the using this command would be '(SET-PROBS 'x)' where x is a chance node. The probabilities must be provided as a list, such as '(.3 .2 .5)'. The probabilities in this list must sum to one.

SET-VALS
Enter: (SET-VALS 'value-or-det-node-name)
Sets the values for a value or a deterministic node. This command will query the user to provide the values for the deterministic node. You must have all predecessors fully defined including the outcomes or alternatives for each node. An example of its use would be '(SET-VALS 'v)' where v is a deterministic node which could be a value node. The values are provided as a single number, such as '4.5'.

REDUCTION COMMANDS

The reduction commands are for reducing or analyzing the influence diagram. These commands must be in a specific sequence. Solution of the influence diagram should follow the algorithm set down in the article "Evaluating Influence Diagrams", by Ross D. Shachter, published in Operations Research, Volume No. 34 (November-December 1986).

BAYES
Enter: (BAYES 'node-name-arc-origin 'node-name-successor)
Reverses the arc between two chance nodes. The command "(BAYES 'x 'y)" reverses the arc from chance node x to chance node y using Bayes' Theorem. The chance node x must be a predecessor of y.

EXPEC
Enter: (EXPEC 'value-node-name 'chance-node-name)
Takes the expectation of the value node with repect to a chance node. The command "(EXPEC 'v 'x)" takes the expectation of value node v with respect to chance node x.

INTEGRATE-OUT
Enter: (INTEGRATE-OUT 'chance-node-name)
Integrates the probability distribution of a chance node into the successor and removes the chance node. The command "(INTEGRATE-OUT 'x)" removes chance node x from the influence diagram by forming the joint of x and its SINGLE successor and then integrating out x (actually summing out and producing the marginal probability distribution).

MAXIM
Enter: (MAXIM 'value-node-name 'decision-node-name)
Maximizes the value node with respect to a decision node. The command "(MAXIM 'v 'd)" maximizes the value node v with respect to decision node d. After maximization, the decision node will remain in the influence diagram and have node kind decn-det.

REDUCE-DET
Enter: (REDUCE-DET 'det-node-name)
Removes the deterministic node predecessors of a deterministic node which could be a value node. The command "(REDUCE-DET 'v)" will remove the deterministic predecessors of deterministic node v.

SHOW COMMANDS

The show commands are for displays of information to the user. These commands will either show the user what stage of development the influence diagram is in or show the information for a particular node.

SHOW-D*
Enter: (SHOW-D* 'decn-det-node-name)
Prints to the screen the optimal alternative for the decn-det node d. The command '(SHOW-D* 'd)' will show the optimal alternative for the decision node d after it has been removed by maximizing the value node. After maximization, the decision node will remain in the influence diagram and have node kind decn-det.

SHOW-ID
Enter: (SHOW-ID)
Prints to the screen the current influence diagram. The information displayed is the entire influence diagram.

SHOW-GRAPH
Enter: (SHOW-GRAPH)
Prints to the screen the graph structure of the current influence diagram. The graph structure consists of all nodes with their name, kind, predecessors, and successors. This information is the same information represented by a drawing (or graph) of the influence diagram.

SHOW-NODE
Enter: (SHOW-NODE 'node-name)
Prints to the screen the information for a node. The command '(SHOW-NODE 'x)' will show the information for the node x.

SHOW-PROBS
Enter: (SHOW-PROBS 'chance-node-name)
Prints to the screen the probabilities for a chance node. The command '(SHOW-PROBS 'x)' will show the probabilities for chance node x.

SHOW-VALS
Enter: (SHOW-VALS 'value-or-det-node-name)
Prints to the screen the values for a value or det node. The command '(SHOW-VALS 'v)' will show the values for node v.

SESSION1

This section contains an example of system use created with the transcript option. The problem is a decision regarding the location of a party. The alternatives are outdoors, porch, or indoors. The weather is a factor in the problem, and, since the weather is a random variable, the weather is represent by a chance node. There are two outcomes for weather: sunshine and rain. At the time the decision is made, the true weather is not known. There is however a test on the weather that is available to the decision maker. Therefore, the decision is influenced by the knowledge of test results on the weather. The possible outcomes for the test are the following: test says sunshine (denoted by test-sunshine) and test says rain (denoted by test-rain). The values of the decision maker are influenced by the location and the weather.

The remainder of this section is the transcript file.

```
 Transcript file SESSION1 created in default directory.
 Ready to go to work!
>
> ; NOTE: the comments of SESSION1 are set off with
> ; semicolons.  LISP will ignore the semicolon and
> ; anything that follows.  Use of comments will allow
> ; easier review of the transcript file.
>
> ; To create a new influence diagram use the NEW-ID
> ; command.
>
> (new-id 'party)
  The new influence diagram PARTY has been created.

>
> ; A new influence diagram called PARTY is started.
>
>
> ; Node structure can be input one of two ways: using
> ; SET-NODES or ADD-NODE command.
>
> (set-nodes)

  Node name: > location
  Kind of node: > decision
  List alternatives: > (outdoors porch indoors)
  The DECISION node LOCATION is set in the influence
  diagram.
```

```
  Set another node (Y or N)? > y

  Node name: > weather
  Kind of node: > chance
  List outcomes: > (sunshine rain)
  The CHANCE node WEATHER is set in the influence
  diagram.

  Set another node (Y or N)? > y

  Node name: > v
  Kind of node: > value
  The VALUE node V is set in the influence diagram.

  Set another node (Y or N)? > n

>
> ; The ADD-NODE command is demonstrated by adding the
> ; chance node TEST-WEATHER to the id.  This node
> ; represents the test on the weather.
>
> (add-node 'test-weather 'chance)
  The CHANCE node TEST-WEATHER has been added to the
  influence diagram.

>
> ; Notice that the SET-NODES command allowed the input
> ; of the alternatives for the decision node and the
> ; outcomes for the chance node.  The SET-NODES allows
> ; one node to be added as well as several nodes.  The
> ; ADD-NODE command is quicker to add a single
> ; deterministic or value node, but in the example a
> ; chance node was added.  The chance node TEST-WEATHER
> ; requires its outcomes, and this operation is done
> ; with the SET-OUTCOMES command.
>
> (set-outcomes 'test-weather '(test-sunshine test-rain))
  The outcomes have been set for chance node TEST-WEATHER.

>
> ; Individual node information is displayed with the
> ; SHOW-NODE command.  Individual node information is
> ; convenient when checking the node development during
> ; the input of information.
>
> ; All nodes of the PARTY id exist, and to check the
> ; information use the SHOW-ID command.  This command
> ; shows the entire influence diagram.
>
```

```
> (show-id)

                    Influence  Diagram  PARTY

*****   Node:  LOCATION   *****
KIND: DECISION
ALTS: (OUTDOORS PORCH INDOORS)
DIM: 3

*****   Node:  WEATHER   *****
KIND: CHANCE
OUTCOMES: (SUNSHINE RAIN)
DIM: 2

*****   Node:  V   *****
KIND: VALUE

*****   Node:  TEST-WEATHER   *****
KIND: CHANCE
OUTCOMES: (TEST-SUNSHINE TEST-RAIN)
DIM: 2

>
>
>
> ; Next, the relationship between the nodes is defined
> ; by input of predecessor information for each node.
> ; For this process, the SET-PREDS command is used.
>
> (set-preds 'location '(test-weather))
  The predecessors of node LOCATION have been set.

>
>
> (set-preds 'test-weather '(weather))
  The predecessors of node TEST-WEATHER have been set.

>
>
> (set-preds 'v '(location weather))
  The predecessors of node V have been set.

>
>
> ; Once the predecessors of each node are set, the
> ; arcs of the id are defined.  Each node has a list of
> ; predecessors and successors.  The absence of either
> ; predecessors or successors is indicated by the word
> ; NIL.  To see this graph structure of the id, use the
> ; SHOW-GRAPH command.  PerForma does not have graphics
> ; capability, so the graph structure is given in text
> ; format.  The id must be drawn by hand using this
> ; information.
>
```

```
> (show-graph)

                  Graph  Structure  of  PARTY

Node: LOCATION     Kind: DECISION
Predecessors: (TEST-WEATHER)
Successors: (V)

Node: WEATHER     Kind: CHANCE
Predecessors: NIL
Successors: (V TEST-WEATHER)

Node: V     Kind: VALUE
Predecessors: (LOCATION WEATHER)
Successors: NIL

Node: TEST-WEATHER     Kind: CHANCE
Predecessors: (WEATHER)
Successors: (LOCATION)

>
>
> ; Once the graph structure of the id is set, further
> ; information is input into the nodes.  The chance nodes
> ; require the probabilities for the outcomes.  The value
> ; node requires the values.
>
> ; To input the probabilities for a chance node, they must
> ; be in list format (i.e., probabilities must be in
> ; parentheses).  There is a .4 probability of sunshine
> ; and a .6 probability of rain.  The entry for the
> ; WEATHER probabilities is (.4 .6).
>
> (set-probs 'weather)

No predecessors exist.  Enter the probabilities for this
node's outcomes.

Outcomes: (SUNSHINE RAIN)
Probs:  > (.4 .6)

  The probabilities for chance node WEATHER have been set.

>
> ; The TEST-WEATHER probabilities are entered next.  The
> ; accuracy of the test is 80 percent.  So if the weather
> ; is sunshine, then the test outcome will be TEST-SUNSHINE
> ; with a probability of .8 and TEST-RAIN with a .2
> ; probability.  Input of this information is as follows:
>
```

```
> (set-probs 'test-weather)

Given the single predecessor's alternative/outcome,
enter the probabilities for this node's outcomes.

PREDECESSOR : OUTCOMES for TEST-WEATHER
(WEATHER) : (TEST-SUNSHINE TEST-RAIN)

(SUNSHINE) : > (.8 .2)
(RAIN) : > (.2 .8)

  The probabilities for chance node TEST-WEATHER have been
  set.

>
> ; To set the value for the value node, use the SET-VALS
> ; command.  The individual values are input using a
> ; single value not in parentheses.
>
> (set-vals 'v)

Given the predecessors' alternatives/outcomes,
enter the value for this node.

PREDECESSORS : VALUES for V
(LOCATION WEATHER)

(OUTDOORS SUNSHINE) : > 100
(OUTDOORS RAIN) : > 0
(PORCH SUNSHINE) : > 90
(PORCH RAIN) : > 20
(INDOORS SUNSHINE) : > 40
(INDOORS RAIN) : > 50

  The values have been set for the value node V.

>
>
> ; Check the information just input with SHOW-ID command.
>
> (show-id)

                    Influence  Diagram  PARTY

*****   Node:  LOCATION   *****
KIND: DECISION
ALTS: (OUTDOORS PORCH INDOORS)
DIM: 3
PREDS: (TEST-WEATHER)
SUCCS: (V)
```

```
*****   Node:  WEATHER   *****
KIND: CHANCE
OUTCOMES: (SUNSHINE RAIN)
DIM: 2
SUCCS: (V TEST-WEATHER)
PROBS: array #2
PREDS: NIL

*****   Node:  V   *****
KIND: VALUE
PREDS: (LOCATION WEATHER)
VALS: array #4


  Do you want more (Y or N)? > y

*****   Node:  TEST-WEATHER   *****
KIND: CHANCE
OUTCOMES: (TEST-SUNSHINE TEST-RAIN)
DIM: 2
SUCCS: (LOCATION)
PREDS: (WEATHER)
PROBS: array #3

>
> ; The probabilities are indicated by the PROBS in the
> ; chance node information.  The values are indicated by
> ; the VALS in the value node information.
>
> ; At this point, the influence diagram is built.  Store
> ; the id with the STORE-ID command which will create a
> ; file in default directory.  This file can be loaded
> ; later for further analysis using the LOAD-ID command.
>
> (store-id 'party)
  The influence diagram is stored under the file name
  PARTY.lsp in the default directory.

>
> ; The evaluation algorithm is applied to the id to solve
> ; for the optimal solution.
>
> ; In applying the algorithm, the first step is to reverse
> ; the arc from weather to test-weather.  The arc reversal
> ; is done with the BAYES command (applies Bayes' Rule).
>
> (bayes 'weather 'test-weather)
  The conditioning of node TEST-WEATHER upon node WEATHER
  has been reversed using Bayes' Rule.

>
```

```
> ; The operation of the arc reversal will change the
> ; predecessors and successors of the two chance nodes.
> ; This information is checked most easily with
> ; SHOW-GRAPH.
>
> (show-graph)

                  Graph  Structure  of  PARTY

Node: LOCATION     Kind: DECISION
Predecessors: (TEST-WEATHER)
Successors: (V)

Node: WEATHER     Kind: CHANCE
Predecessors: (TEST-WEATHER)
Successors: (V)

Node: V     Kind: VALUE
Predecessors: (LOCATION WEATHER)
Successors: NIL

Node: TEST-WEATHER     Kind: CHANCE
Predecessors: NIL
Successors: (WEATHER LOCATION)

>
> ; Notice that TEST-WEATHER is now a predecessor of
> ; WEATHER.  The probabilities have changed for both
> ; chance nodes.  The new probabilities for the node
> ; WEATHER are displayed with the SHOW-PROBS command.
>
> (show-probs 'weather)

       Probabilities  for  Node  WEATHER

The probabilities depend on the alternative/outcome of its
predecessor.

PREDECESSOR : OUTCOMES for WEATHER
(TEST-WEATHER) : (SUNSHINE RAIN)

(TEST-SUNSHINE) : (0.727273 0.272727)
(TEST-RAIN) : (0.142857 0.857143)

>
> ; The new probabilities for TEST-WEATHER are displayed
> ; using the same command.  Note that the display format
> ; is slightly different since no predecessor exists.
>
```

```
> (show-probs 'test-weather)

       Probabilities  for  Node  TEST-WEATHER

Since no predecessors exist, the probabilities are the
following:

Outcomes: (TEST-SUNSHINE TEST-RAIN)
Probs: (0.440000 0.560000)

_
>
> ; The next step in the evaluation procedure is the chance
> ; node removal of WEATHER.  This operation is known as
> ; taking the expectation of the value node V with respect
> ; to the chance node WEATHER.  The command used is EXPEC
> ; command.
>
> (expec 'v 'weather)
  Node WEATHER has been removed by expectation.

_
>
> ; Use the SHOW-VALS command to display the changed values
> ; in the value node V.
>
> (show-vals 'v)

       Values  for  Node  V

The value depends on the alternatives/outcomes of its
predecessors.

PREDECESSORS : VALUES for V
(LOCATION TEST-WEATHER)

(OUTDOORS TEST-SUNSHINE) : 72.7273
(OUTDOORS TEST-RAIN) : 14.2857
(PORCH TEST-SUNSHINE) : 70.9091
(PORCH TEST-RAIN) : 30.0000
(INDOORS TEST-SUNSHINE) : 42.7273
(INDOORS TEST-RAIN) : 48.5714

_
>
> ; The next step in the algorithm is the decision node
> ; removal of LOCATION by taking the maximization of V
> ; with respect to LOCATION.
>
> (maxim 'v 'location)
  Node LOCATION has been removed by maximization.

_
>
```

```
> (show-id)

                    Influence  Diagram  PARTY

*****   Node:  LOCATION   *****
KIND: DECN-DET
ALTS: (OUTDOORS PORCH INDOORS)
DIM: 3
PREDS: (TEST-WEATHER)
SUCCS: NIL
VALS: array #13

*****   Node:  V   *****
KIND: VALUE
PREDS: (TEST-WEATHER)
VALS: array #12

*****   Node:  TEST-WEATHER   *****
KIND: CHANCE
OUTCOMES: (TEST-SUNSHINE TEST-RAIN)
DIM: 2
SUCCS: (V)
PREDS: NIL
PROBS: array #7

>
> ; The value node information is changed.
>
> (show-vals 'v)

       Values  for  Node  V

The value depends on the alternative/outcome of its
predecessor.

PREDECESSOR : VALUES for V
(TEST-WEATHER)

(TEST-SUNSHINE) : 72.7273
(TEST-RAIN) : 48.5714

>
> ; There is one more node to remove from the id.  To
> ; remove the node TEST-WEATHER, the EXPEC command is
> ; used.
>
> (expec 'v 'test-weather)
  Node TEST-WEATHER has been removed by expectation.

>
```

```
> (show-id)

                    Influence  Diagram  PARTY

*****   Node:  LOCATION   *****
KIND: DECN-DET
ALTS: (OUTDOORS PORCH INDOORS)
DIM: 3
PREDS: (TEST-WEATHER)
SUCCS: NIL
VALS: array #13

*****   Node:  V   *****
KIND: VALUE
PREDS: NIL
VALS: array #15

>
> ; The influence diagram's expected value is now captured
> ; in the value node since this node has no predecessors.
>
> (show-vals 'v)

       Value  for  Node  V

Since no predecessors exist, the value is the following:
Expected value: 59.2000

>
> ; The node LOCATION has been transformed into a node
> ; that has the node kind of decision-deterministic
> ; (i.e. DECN-DET).  This DECN-DET node tells us the
> ; optimal alternative in making the decision for the
> ; location of the party.  Optimal strategy is displayed
> ; with the SHOW-D* command.
>
> (show-d* 'location)

       Optimal  Alternatives  for  Node  LOCATION

The optimal alternative depends on the alternative/outcome
of its predecessor.

PREDECESSOR : OPTIMAL ALTERNATIVES for LOCATION
(TEST-WEATHER)

(TEST-SUNSHINE) : OUTDOORS
(TEST-RAIN) : INDOORS

>
```

```
> ; The optimal alternatives for the location are
> ; conditioned on the outcomes of TEST-WEATHER.  The
> ; optimal strategy is the following:
> ; -- If the TEST on WEATHER says SUNSHINE, then the
> ;    location should be outdoors.
> ; -- If the TEST says RAIN, then the location should be
> ;    indoors.
>
> ; To stop the system execution, use the EXIT command.
>
> (exit)
```

Bibliography

1. ARBORIST Decision Tree Software. Texas Instruments Incorporated, 1985.

2. DAVID, Influence Diagram Processing System for the Macintosh. Ross D. Shachter and the Center for Health Policy Research and Education, Duke University, 1987.

3. Hansen, Wilfred J. "User Engineering Principles for Interactive Systems," Proceedings of the Fall Joint Computer Conference, 39, AFIPS Press, Montvale, NJ, 1971.

4. Howard, Ronald A. "The Evolution of Decision Analysis," The Principles and Applications of Decision Analysis, Volume I, edited by Ronald A. Howard and James E. Matheson. Strategic Decisions Group, Menlo Park, Calif., 1984.

5. Howard, Ronald A., and James E. Matheson. "Influence Diagrams," The Principles and Applications of Decision Analysis, Volume II, edited by Ronald A. Howard and James E. Matheson. Strategic Decisions Group, Menlo Park, Calif., 1984.

6. Kenley, C. R. Influence Diagram Models With Continuous Variables. PhD dissertation. Stanford University, Calif., June 1986.

7. Newell, Allan, and Herbert Simon. Human Problem Solving, Englewood Cliffs, NJ; Prentice-Hall, 1972.

8. Norman, Donald A. "Stages and Levels in Human-Machine Interaction," International Journal of Man-Machine Studies 21, 1984.

9. Owen, Daniel L. "The Use of Influence Diagrams in Structuring Complex Decision Problems," The Principles and Applications of Decision Analysis, Volume II, edited by Ronald A. Howard and James E. Matheson. Strategic Decisions Group, Menlo Park, Calif., 1984.

10. Shachter, Ross D. "Evaluating Influence Diagrams," Operations Research, 34: 871-882 (November-December 1986).

11. Smith, Sid L., and Jane N. Mosier. Design Guidelines for the User Interface for Computer-Based Information Systems, MITRE Corporation, Bedford, Mass., Electronic Systems Division, 1984.

12. SMLTREE, The All Purpose Decision Tree Builder, by Jim Hollenburg and the Pratt Medical Group, 1986.

13. Tatman, Joseph A. Decision Processes in Influence Diagrams: Formulation and Analysis. PhD dissertation. Department of Engineering-Economic Systems, Stanford University, Calif., December 1985.

VITA

Captain Thomas M. Burwell was born on 26 May 1956 in Dhahran, Saudi Arabia. He graduated from high school at Notre Dame International School in Rome, Italy, in 1974 and attended the University of Georgia, from which he received the degree of Bachelor of Science in Mathematics in June 1978. Upon graduation, he was awarded the designation of AFROTC Distinguished Graduate and received a commission in the USAF. He completed pilot training and received his wings in May 1980. He served as a C-141 pilot and flight instructor in the 20th Military Airlift Squadron, Charleston AFB, South Carolina, until April 1984. He was then assigned as C-141 flight test pilot for the Warner Robins Air Logistics Center, Robins AFB, Georgia, until entering the graduate Operations Research program in the School of Engineering, Air Force Institute of Technology, in May 1986.

Permanent address: 1205 Windsong Trail
Richardson, Texas 75081

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/MA/87D-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENC	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
See Block 19

12. PERSONAL AUTHOR(S)
Thomas M. Burwell, B.S., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1987 December	82

16. SUPPLEMENTARY NOTATION
Fr. Luck

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	04		Decision Making, Decision Theory, Mathematics, Operations Research, Probability, Computer Programs.
12	03		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

TITLE: PERFORMA -- A Personel Influence Diagram System for Decision Analysis

Thesis Chairman: Joseph A. Tatman, Captain, USAF
Associate Professor of Mathematics

Abstract: See back.

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Joseph A. Tatman, Captain, USAF	22b. TELEPHONE (Include Area Code) (513) 255-3098
	22c. OFFICE SYMBOL AFIT/ENC

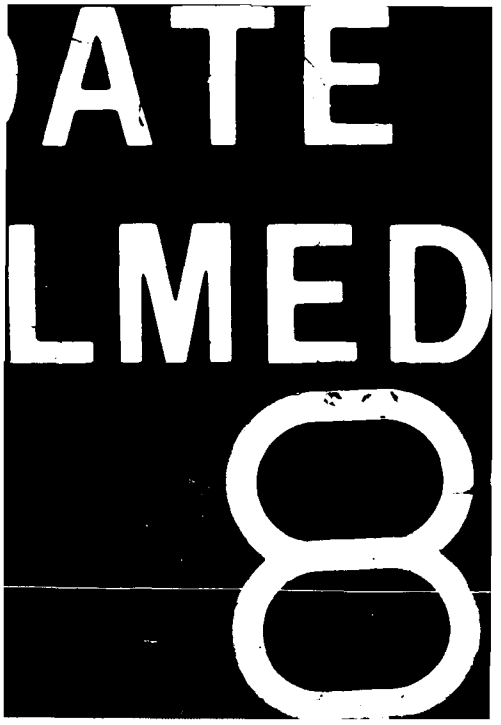
DD FORM 1473, 84 MAR — Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ABSTRACT

The major goal of this research is to develop a widely accessible software system for analyzing and solving influence diagrams for the decision analysis community at the U.S. Air Force Institute of Technology.

A software system for influence diagram analysis, PerForma, was developed. The system makes the influence diagram, a powerful modeling framework for both formulation and analysis, easily accessible. XLISP Version 1.7 was used to write the system. The Amiga, Atari, Macintosh, IBM PC and PC compatible computers have a fully operational system. The system is cost free and has no distribution restrictions expect for commercial use. The user's manual was written with a documented session of system use. A tutorial was written explaining influence diagram modeling and solution procedure. PerForma was validated through its use in decision analysis application theses.


ATE
LMED
8